Task 1:

(tensorflow-gpu) C:\Users\Alberto>conda info

Current conda install:

```
             platform : win-64
        conda version : 4.3.23
       conda is private : False
      conda-env version : 4.3.23
    conda-build version : not installed
         python version : 3.6.2.final.0
       requests version : 2.14.2
       root environment : D:\deeplearning\anaconda  (writable)
    default environment : D:\deeplearning\anaconda\envs\tensorflow-gpu
       envs directories : D:\deeplearning\anaconda\envs
                          C:\Users\Alberto\AppData\Local\conda\conda\envs
                          C:\Users\Alberto\.conda\envs
          package cache : D:\deeplearning\anaconda\pkgs
                          C:\Users\Alberto\AppData\Local\conda\conda\pkgs
           channel URLs : https://repo.continuum.io/pkgs/free/win-64
                          https://repo.continuum.io/pkgs/free/noarch
                          https://repo.continuum.io/pkgs/r/win-64
                          https://repo.continuum.io/pkgs/r/noarch
                          https://repo.continuum.io/pkgs/pro/win-64
                          https://repo.continuum.io/pkgs/pro/noarch
                          https://repo.continuum.io/pkgs/msys2/win-64
                          https://repo.continuum.io/pkgs/msys2/noarch
            config file : None
             netrc file : None
           offline mode : False
             user-agent : conda/4.3.23 requests/2.14.2 CPython/3.6.2 Windows/10 Windows/10.0.15063
          administrator : False
```

Task 2:

In [75]:
```
import numpy as np
import scipy.linalg
```

In [76]:
```
a = np.array([2, 4, 6])
b = np.array([8, 10, 12])
c = np.array([1, 3, 5])
d = np.array([7, 9, 11])
print(a)
print(b)
print(c)
print(d)
```

```
[2 4 6]
[ 8 10 12]
[1 3 5]
[ 7  9 11]
```

In [77]:
```
a.ndim
```

Out[77]: 1

In [78]:
```
a.size
```

Out[78]: 3

In [79]:
```
a.shape
```

Out[79]: (3,)

In [80]:
```
a.shape[1-1]
```

Out[80]: 3

In [81]:
```
np.array([[1., 2., 3.], [4., 5., 6.]])
```

Out[81]:
```
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.]])
```

```
a = np.vstack([np.hstack([c,d]), np.hstack([a,b]), np.hstack([13, 15, 17, 19,
21, 23]),
               np.hstack([14, 16, 18, 20, 22, 24]), np.hstack([25, 27, 29, 31,
 33, 35]),
              np.hstack(         30, 32, 34, 36])])
```

In [84]:
```
print(a)
a[1,4]
```

```
[[ 1  3  5  7  9 11]
 [ 2  4  6  8 10 12]
 [13 15 17 19 21 23]
 [14 16 18 20 22 24]
 [25 27 29 31 33 35]
 [26 28 30 32 34 36]]
```

Out[84]: 10

In [87]:
```
a[1,:]
```

Out[87]: array([2, 4, 6, 8, 10, 12])

In [89]:
```
print(a)
a[0:5,:]
```

```
[[ 1  3  5  7  9 11]
 [ 2  4  6  8 10 12]
 [13 15 17 19 21 23]
 [14 16 18 20 22 24]
 [25 27 29 31 33 35]
 [26 28 30 32 34 36]]
```

Out[89]:
```
array([[ 1,  3,  5,  7,  9, 11],
       [ 2,  4,  6,  8, 10, 12],
       [13, 15, 17, 19, 21, 23],
       [14, 16, 18, 20, 22, 24],
       [25, 27, 29, 31, 33, 35]])
```

In [94]:
```
a[-5:]
```

Out[94]:
```
array([[ 2,  4,  6,  8, 10, 12],
       [13, 15, 17, 19, 21, 23],
       [14, 16, 18, 20, 22, 24],
       [25, 27, 29, 31, 33, 35],
       [26, 28, 30, 32, 34, 36]])
```

In [95]:
```
a[0:3][:,3:5]
```

Out[95]:
```
array([[ 7,  9],
       [ 8, 10],
       [19, 21]])
```

In [97]:
```
a[np.ix_([1,3,4],[0,2])]
```

Out[97]:
```
array([[ 2,  6],
       [14, 18],
       [25, 29]])
```

In [98]:
```
a[ 2:21:2,:]
```

Out[98]:
```
array([[13, 15, 17, 19, 21, 23],
       [25, 27, 29, 31, 33, 35]])
```

In [99]:
```
a[ ::2,:]
```

Out[99]:
```
array([[ 1,  3,  5,  7,  9, 11],
       [13, 15, 17, 19, 21, 23],
       [25, 27, 29, 31, 33, 35]])
```

In [100]:
```
a[ ::-1,:]
```

Out[100]:
```
array([[26, 28, 30, 32, 34, 36],
       [25, 27, 29, 31, 33, 35],
       [14, 16, 18, 20, 22, 24],
       [13, 15, 17, 19, 21, 23],
       [ 2,  4,  6,  8, 10, 12],
       [ 1,  3,  5,  7,  9, 11]])
```

In [101]:
```
a[np.r_[:len(a),0]]
```

Out[101]:
```
array([[ 1,  3,  5,  7,  9, 11],
       [ 2,  4,  6,  8, 10, 12],
       [13, 15, 17, 19, 21, 23],
       [14, 16, 18, 20, 22, 24],
       [25, 27, 29, 31, 33, 35],
       [26, 28, 30, 32, 34, 36],
       [ 1,  3,  5,  7,  9, 11]])
```

In [102]:
```
print(a)
a.T
```

```
[[ 1  3  5  7  9 11]
 [ 2  4  6  8 10 12]
 [13 15 17 19 21 23]
 [14 16 18 20 22 24]
 [25 27 29 31 33 35]
 [26 28 30 32 34 36]]
```

Out[102]:
```
array([[ 1,  2, 13, 14, 25, 26],
       [ 3,  4, 15, 16, 27, 28],
       [ 5,  6, 17, 18, 29, 30],
       [ 7,  8, 19, 20, 31, 32],
       [ 9, 10, 21, 22, 33, 34],
       [11, 12, 23, 24, 35, 36]])
```

In [103]:
```
a.conj().T
```

Out[103]:
```
array([[ 1,  2, 13, 14, 25, 26],
       [ 3,  4, 15, 16, 27, 28],
       [ 5,  6, 17, 18, 29, 30],
       [ 7,  8, 19, 20, 31, 32],
       [ 9, 10, 21, 22, 33, 34],
       [11, 12, 23, 24, 35, 36]])
```

In [113]:
```
a = np.random.rand(3,2)
b = np.random.rand(2,4)
print(a)
print(b)
a.dot(b)
```

```
[[ 0.98959547  0.37835622]
 [ 0.15888081  0.79275885]
 [ 0.86752877  0.5405667 ]]
[[ 0.61508245  0.34311348  0.1601979   0.71695496]
 [ 0.10415973  0.64712419  0.67203502  0.58146594]]
```

Out[113]:
```
array([[ 0.64809229,  0.58438701,  0.41279974,  0.92949664],
       [ 0.18029835,  0.56752758,  0.55821408,  0.57487266],
       [ 0.589907  ,  0.6474746 ,  0.50225604,  0.93630017]])
```

In [116]:
```
b = np.random.rand(3,2)
print(b)
a*b
```

```
[[ 0.40427099  0.04550985]
 [ 0.89131055  0.45224205]
 [ 0.01552946  0.00896352]]
```

Out[116]:
```
array([[ 0.40006474,  0.01721893],
       [ 0.14161214,  0.35851889],
       [ 0.01347225,  0.00484538]])
```

In [117]:
```
a/b
```

Out[117]:
```
array([[  2.44785182,   8.31372189],
       [  0.17825528,   1.75295254],
       [ 55.86342323,  60.30743683]])
```

In [118]:
```
a**3
```

Out[118]:
```
array([[ 0.96911006,  0.05416299],
       [ 0.00401065,  0.49822246],
       [ 0.65290749,  0.15796027]])
```

In [119]:
```
(a>0.5)
```

Out[119]:
```
array([[ True, False],
       [False,  True],
       [ True,  True]], dtype=bool)
```

In [121]:
```
np.nonzero(a>.5)
```

Out[121]:
```
(array([0, 1, 2, 2], dtype=int64), array([0, 1, 0, 1], dtype=int64))
```

In [131]:
```
v = np.random.rand(1,2)
print(a)
print(v)
a[:,np.nonzero(v>0.5)[0]]
```

```
[[ 0.98959547  0.37835622]
 [ 0.15888081  0.79275885]
 [ 0.86752877  0.5405667 ]]
[[ 0.26184635  0.93517247]]
```

Out[131]:
```
array([[ 0.98959547],
       [ 0.15888081],
       [ 0.86752877]])
```

In [140]:
```
v = np.random.rand(2,1)

print(v.T)
#a[:,v.T>0.5]
```

```
[[ 0.6552968   0.52337433]]
```

In [143]:
```
a[a<0.5]=0
print(a)
```

```
[[ 0.98959547  0.        ]
 [ 0.          0.79275885]
 [ 0.86752877  0.5405667 ]]
```

In [145]:
```
a[:] = 3
print(a)
```

```
[[ 3.  3.]
 [ 3.  3.]
 [ 3.  3.]]
```

In [147]:
```
y = a.copy()
print(y)
```

```
[[ 3.  3.]
 [ 3.  3.]
 [ 3.  3.]]
```

In [149]:
```
y = a[1,:].copy()
print(y)
```

```
[ 3.  3.]
```

In [151]:
```
y = a.flatten()
print(y)
```

```
[ 3.  3.  3.  3.  3.  3.]
```

In [153]:
```
np.arange(1.,11.)
```

Out[153]:
```
array([  1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.])
```

In [154]:
```
np.arange(10.)
```

Out[154]:
```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.])
```

In [155]:
```
np.arange(1.,11.)[:, np.newaxis]
```

Out[155]:
```
array([[  1.],
       [  2.],
       [  3.],
       [  4.],
       [  5.],
       [  6.],
       [  7.],
       [  8.],
       [  9.],
       [ 10.]])
```

In [156]:
```
np.zeros((3,4))
```

Out[156]:
```
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

In [157]:
```
np.     zeros((3,4,5))
```

Out[157]:
```
array([[[ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.]],

       [[ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.]],

       [[ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.],
        [ 0.,  0.,  0.,  0.,  0.]]])
```

In [159]:
```
np.ones((3,4))
```

Out[159]:
```
array([[ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.]])
```

In [160]:
```
np.eye(3)
```

Out[160]:
```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

In [163]:
```
a = np.random.rand(4,4)
np.diag(a)
```

Out[163]:
```
array([ 0.76489529,  0.8006363 ,  0.34092449,  0.54093854])
```

In [164]:
```
np.random.rand(3,4)
```

Out[164]:
```
array([[ 0.21478567,  0.36995484,  0.66877802,  0.13285564],
       [ 0.98349762,  0.4395292 ,  0.56941271,  0.6906989 ],
       [ 0.62955289,  0.68399971,  0.4319856 ,  0.63442975]])
```

In [165]:
```
np.linspace(1,3,4)
```

Out[165]:
```
array([ 1.        ,  1.66666667,  2.33333333,  3.        ])
```

In [167]:
```
np.mgrid[0:9.,0:6.]
```

Out[167]:
```
array([[[ 0.,  0.,  0.,  0.,  0.,  0.],
        [ 1.,  1.,  1.,  1.,  1.,  1.],
        [ 2.,  2.,  2.,  2.,  2.,  2.],
        [ 3.,  3.,  3.,  3.,  3.,  3.],
        [ 4.,  4.,  4.,  4.,  4.,  4.],
        [ 5.,  5.,  5.,  5.,  5.,  5.],
        [ 6.,  6.,  6.,  6.,  6.,  6.],
        [ 7.,  7.,  7.,  7.,  7.,  7.],
        [ 8.,  8.,  8.,  8.,  8.,  8.]],

       [[ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.],
        [ 0.,  1.,  2.,  3.,  4.,  5.]]])
```

In [168]:
```
np.ogrid[0:9.,0:6.]
```

Out[168]:
```
[array([[ 0.],
        [ 1.],
        [ 2.],
        [ 3.],
        [ 4.],
        [ 5.],
        [ 6.],
        [ 7.],
        [ 8.]]), array([[ 0.,  1.,  2.,  3.,  4.,  5.]])]
```

In [169]:
```
np.meshgrid([1,2,4],[2,4,5])
```

Out[169]:
```
[array([[1, 2, 4],
        [1, 2, 4],
        [1, 2, 4]]), array([[2, 2, 2],
        [4, 4, 4],
        [5, 5, 5]])]
```

In [170]:
```
np.ix_([1,2,4],[2,4,5])
```

Out[170]:
```
(array([[1],
        [2],
        [4]]), array([[2, 4, 5]]))
```

In [172]:
```
np.tile(a, (2, 3))
```

Out[172]:
```
array([[ 0.76489529,  0.26067692,  0.85202235,  0.13770249,  0.76489529,
         0.26067692,  0.85202235,  0.13770249,  0.76489529,  0.26067692,
         0.85202235,  0.13770249],
       [ 0.13902171,  0.8006363 ,  0.84893899,  0.66979412,  0.13902171,
         0.8006363 ,  0.84893899,  0.66979412,  0.13902171,  0.8006363 ,
         0.84893899,  0.66979412],
       [ 0.10466229,  0.70806109,  0.34092449,  0.47561977,  0.10466229,
         0.70806109,  0.34092449,  0.47561977,  0.10466229,  0.70806109,
         0.34092449,  0.47561977],
       [ 0.80871269,  0.43429577,  0.30596384,  0.54093854,  0.80871269,
         0.43429577,  0.30596384,  0.54093854,  0.80871269,  0.43429577,
         0.30596384,  0.54093854],
       [ 0.76489529,  0.26067692,  0.85202235,  0.13770249,  0.76489529,
         0.26067692,  0.85202235,  0.13770249,  0.76489529,  0.26067692,
         0.85202235,  0.13770249],
       [ 0.13902171,  0.8006363 ,  0.84893899,  0.66979412,  0.13902171,
         0.8006363 ,  0.84893899,  0.66979412,  0.13902171,  0.8006363 ,
         0.84893899,  0.66979412],
       [ 0.10466229,  0.70806109,  0.34092449,  0.47561977,  0.10466229,
         0.70806109,  0.34092449,  0.47561977,  0.10466229,  0.70806109,
         0.34092449,  0.47561977],
       [ 0.80871269,  0.43429577,  0.30596384,  0.54093854,  0.80871269,
         0.43429577,  0.30596384,  0.54093854,  0.80871269,  0.43429577,
         0.30596384,  0.54093854]])
```

In [177]:
```
a = np.random.rand(1,3)
b = np.random.rand(1,3)
np.hstack((a,b))
```

Out[177]:
```
array([[ 0.49364749,  0.42410544,  0.46992571,  0.13098445,  0.15038774,
         0.31861061]])
```

In [178]:
```
np.concatenate((a,b))
```

Out[178]:
```
array([[ 0.49364749,  0.42410544,  0.46992571],
       [ 0.13098445,  0.15038774,  0.31861061]])
```

In [179]:
```
a.max()
```

Out[179]:
```
0.49364749172772882
```

In [180]:
```
a.max(0)
```

Out[180]:
```
array([ 0.49364749,  0.42410544,  0.46992571])
```

In [181]:
```
a.max(1)
```

Out[181]:
```
array([ 0.49364749])
```

In [183]:
```
np.maximum(a,b)
```

Out[183]:
```
array([[ 0.49364749,  0.42410544,  0.46992571]])
```

In [187]:
```
v = np.random.rand(3,3)
np.sqrt(np.dot(v,v))
```

Out[187]:
```
array([[ 0.7303104 ,  0.32902246,  0.50987886],
       [ 1.25860475,  0.83066891,  1.14234491],
       [ 0.55441768,  0.35144614,  0.69313307]])
```

In [188]:
```
np.logical_and(a,b)
```

Out[188]:
```
array([[ True,  True,  True]], dtype=bool)
```

In [189]:
```
np.logical_or(a,b)
```

Out[189]:
```
array([[ True,  True,  True]], dtype=bool)
```

In [190]:
```
a & b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-190-ed983a5ba089> in <module>()
----> 1 a & b

TypeError: ufunc 'bitwise_and' not supported for the input types, and the inp
uts could not be safely coerced to any supported types according to the casti
ng rule ''safe''
```

In [191]:
```
a | b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-191-9994be362380> in <module>()
----> 1 a | b

TypeError: ufunc 'bitwise_or' not supported for the input types, and the inpu
ts could not be safely coerced to any supported types according to the castin
g rule ''safe''
```

In [194]:
```
a = np.random.rand(3,3)
print(a)
np.linalg.inv(a)
```

```
[[ 0.81598693  0.01396717  0.62533361]
 [ 0.74112635  0.04415695  0.6914919 ]
 [ 0.38394216  0.21860997  0.04116314]]
```

Out[194]:
```
array([[ 5.35845384, -4.88413254,  0.64418826],
       [-8.43097253,  7.40906064,  3.61646486],
       [-5.20469626,  6.20773405, -0.92136577]])
```

In [195]:
```
np.linalg.pinv(a)
```

Out[195]:
```
array([[ 5.35845384, -4.88413254,  0.64418826],
       [-8.43097253,  7.40906064,  3.61646486],
       [-5.20469626,  6.20773405, -0.92136577]])
```

In [196]:
```
np.linalg.matrix_rank(a)
```

Out[196]:
```
3
```

In [198]:
```
b = np.random.rand(3,3)
np.linalg.solve(a,b)
```

Out[198]:
```
array([[ 1.38975659,  1.9894221 , -0.31252267],
       [ 1.29025376,  0.23680687,  1.06901251],
       [-1.28615668, -2.1039995 ,  1.62336906]])
```

In [206]:
```
U, S, Vh = np.linalg.svd(a)
V = Vh.T
print(V)
print(U)
print(S)
```

```
[[-0.78360282  0.42501968 -0.45312789]
 [-0.06078002  0.67342134  0.73675606]
 [-0.61828182 -0.60486525  0.50186216]]
[[-0.69360947 -0.07450375 -0.71648803]
 [-0.68285531 -0.2487145   0.68691318]
 [-0.22937858  0.96570714  0.12163545]]
[ 1.48050313  0.29563977  0.06367834]
```

In [207]:
```
#np.linalg.cholesky(a).T
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-207-84793e9b0675> in <module>()
----> 1 np.linalg.cholesky(a).T

D:\deeplearning\anaconda\envs\tensorflow-gpu\lib\site-packages\numpy\linalg\l
inalg.py in cholesky(a)
    610     t, result_t = _commonType(a)
    611     signature = 'D->D' if isComplexType(t) else 'd->d'
--> 612     r = gufunc(a, signature=signature, extobj=extobj)
    613     return wrap(r.astype(result_t, copy=False))
    614

D:\deeplearning\anaconda\envs\tensorflow-gpu\lib\site-packages\numpy\linalg\l
inalg.py in _raise_linalgerror_nonposdef(err, flag)
     91
     92 def _raise_linalgerror_nonposdef(err, flag):
---> 93     raise LinAlgError("Matrix is not positive definite")
     94
     95 def _raise_linalgerror_eigenvalues_nonconvergence(err, flag):

LinAlgError: Matrix is not positive definite
```

In [209]:
```
D,V = np.linalg.eig(a)
print(D)
print(V)
```

```
[ 1.16432228  0.07155004 -0.33456529]
[[ 0.66375266  0.40867415 -0.38795161]
 [ 0.65831941 -0.78294427 -0.56722696]
 [ 0.35503234 -0.46902421  0.72646206]]
```

In [212]:
```
Q,R = scipy.linalg.qr(a)
print(Q)
print(R)
```

```
[[-0.69905707 -0.32219088 -0.63836686]
 [-0.63492392 -0.13097537  0.76139154]
 [-0.32892375  0.93757052 -0.11300743]]
[[-1.16726797 -0.10970616 -0.88972816]
 [ 0.          0.1946787  -0.25345185]
 [ 0.          0.          0.12265209]]
```

In [213]:
```
#L,U = scipy.linalg.lu(a)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-213-6f2aa08dcebc> in <module>()
----> 1 L,U = scipy.linalg.lu(a)

ValueError: too many values to unpack (expected 2)
```

In [217]:
```
np.fft.fft(a)
```

Out[217]:
```
array([[ 1.45528771+0.j        ,  0.49633654+0.52945887j,
         0.49633654-0.52945887j],
       [ 1.47677520+0.j        ,  0.37330193+0.56060851j,
         0.37330193-0.56060851j],
       [ 0.64371527+0.j        ,  0.25405560-0.15367346j,
         0.25405560+0.15367346j]])
```

In [220]:
```
np.fft.ifft(a)
```

Out[220]:
```
array([[ 0.48509590+0.j        ,  0.16544551-0.17648629j,
         0.16544551+0.17648629j],
       [ 0.49225840+0.j        ,  0.12443398-0.1868695j ,
         0.12443398+0.1868695j ],
       [ 0.21457176+0.j        ,  0.08468520+0.05122449j,
         0.08468520-0.05122449j]])
```

In [234]:
```
a = np.random.randint(100, size=10)
print(a)
print(np.sort(a))
```

```
[19 90 89 54 45 79 25 56 75 24]
[19 24 25 45 54 56 75 79 89 90]
```

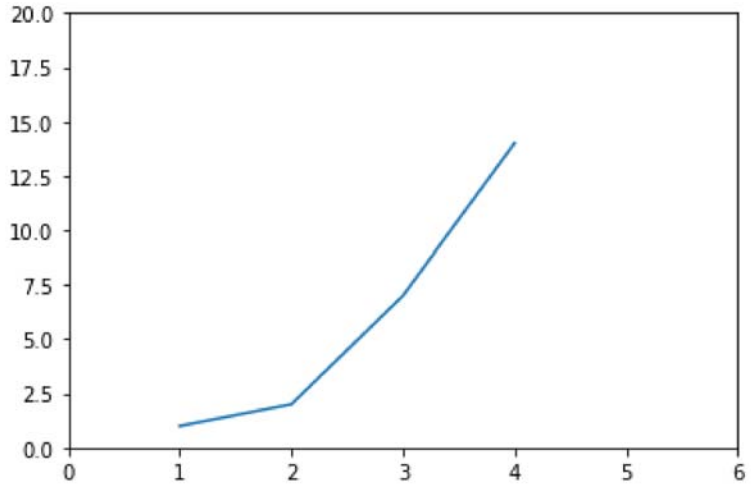
In [238]:
```
a.squeeze()
```

Out[238]:
```
array([19, 90, 89, 54, 45, 79, 25, 56, 75, 24])
```

In [ ]:

Task 3:

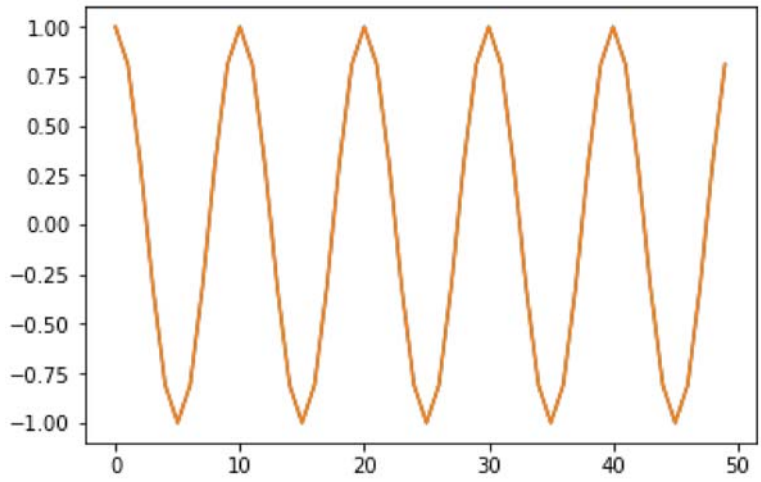
In [1]:
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,2,7,14])
plt.axis([0, 6, 0, 20])
plt.show()
```



Task 4

In [4]:
```
import numpy as np

t1 = np.arange(0.0, 5.0, 0.1)
plt.plot(np.cos(2*np.pi*t1))
plt.show()
```



In [ ]:
```
```

Alberto Fung
COMP576
Assignment0

Task 5:

https://github.com/afung01

Task 6:

https://github.com/afung01/Assignment0